EFFICIENT RECODING USING FAM

[1]C.Gouthami Asst, [2]P.V Kusuma
[1] Assistant Professor, [2]Assistant Professor
[1] Krishna Murthy Engineering College,HYD
[2] JNTUCEP,Pulivendula Andhra Pradesh

Abstract

Complex arithmetic operations are widely used in Digital Signal Processing (DSP) applications. In this work, we focus on optimizing the design of the fused Add-Multiply (FAM) Operator for increasing performance. We investigate techniques to implement the direct recoding of the sum of two numbers in its Modified Booth (MB) form. We introduce a structured and efficient recoding technique and explore three different schemes by incorporating them in FAM designs. In proposed, we focus on Multiply unit which implement the operation. The conventional design of the operator requires complex algothmic process. The drawback of using an this multiplier is that it inserts a significant delay in the critical path of the AM. As there are carry signals to be propagated inside the adder, the critical path depends on the bit-width of the inputs. In order to decrease this delay, a SPST adder can be used which, however, the increases the area occupation and the power dissipation. Comparing them with the FAM designs which use existing recoding schemes, the proposed technique yields considerable reductions in terms of critical delay, hardware complexity of the FAM unit by using Modified Wallace Tree Multiplication. The FAM Architecture is implemented by Verilog Hardware Description Language and it is synthesized by Xilinx tool.

Key Word : FAM, MAD, Booth Recoder

1. INTRODUCTION

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, Subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

The basic multiplication principle is two fold, i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive Addition's of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate bit of the 'multiplicand'. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned numbers, a convenient number system would be the representation of numbers in two's complement format.

2. IMPLEMENTATION OF MULTI OPERAND BOOTH MULTIPLER

2.1 DESIGN

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations.

Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

Modified Booth (MB) is a prevalent form used in multiplication. It is a redundant signed-digit radix-4 en-coding technique. Its main advantage is that it reduces by half the number of partial products in multiplication comparing to any other radix-2 representation.

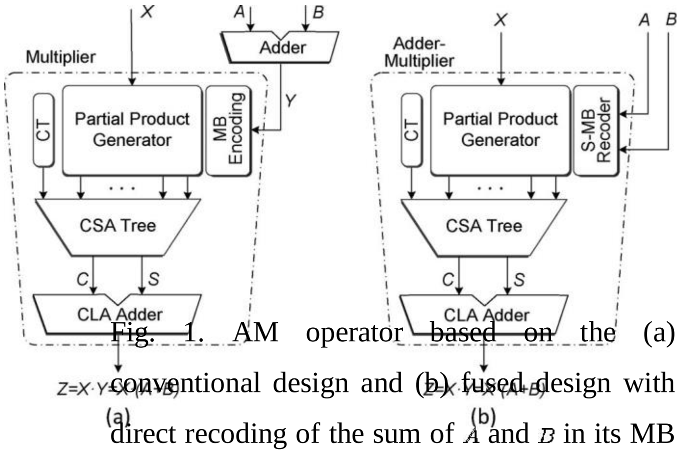


Fig. 1. AM operator based on the (a) conventional design and (b) fused design with direct recoding of the sum of *A* and *B* in its MB representation. The mul-tiplier is a basic parallel multiplier based on the MB algorithm. The terms CT, CSA Tree and CSLA Adder are referred to the Correction Term, the Carry-Sellect Adder Tree and the final Carry-Look-Ahead Adder of the multiplier.

2.2 Circuit Design Features

One of the most advanced types of MAC for general-purpose digital signal processing has been proposed by Elguibaly. It is an architecture in which accumulation has been combined with the carry save adder (CSA) tree that compresses partial products. In the architecture proposed in, the critical path was reduced by eliminating the adder for accumulation and decreasing the number of input bits in the final adder. While it has a better performance because of the reduced critical path compared to the previous VMFU architectures, there is a need to improve the output rate due to the use of the final adder results for accumulation. The architecture to merge the adder block to the accumulator register in the VMFU operator was proposed to provide the possibility of using two separate N/2-bit adders instead of one-bit adder to accumulate the MAC results. Recently, Zicari proposed an architecture that took a merging technique to fully utilize the 4–2 compressor .It also took this compressor as the basic building blocks for the multiplication circuit.

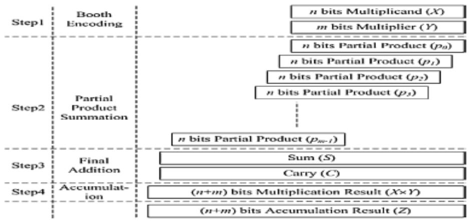
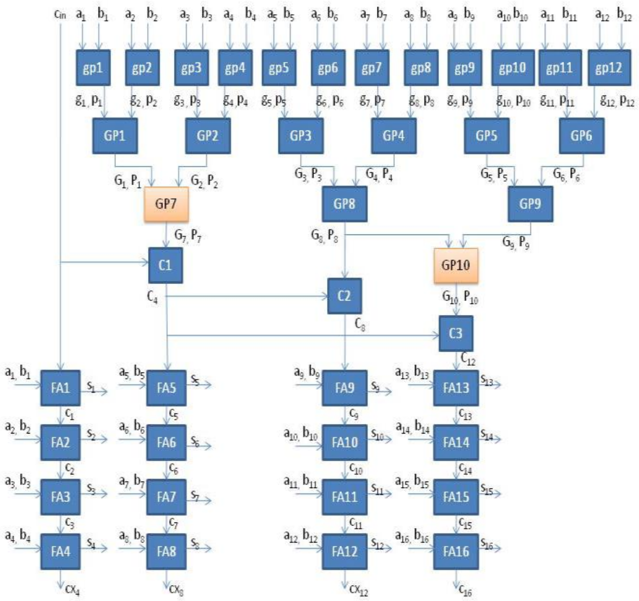

Step1
Booth Encoding
Step2
Partial Product Summation
Step3
Final Addition
Step4
Accumulat-ion
n bits Multiplicand (*X*)
m bits Multiplier (*Y*)
n bits Partial Product (p_0)
n bits Partial Product (p_1)
n bits Partial Product (p_2)
n bits Partial Product (p_3)
n bits Partial Product (p_{m-1})
Sum (*S*)
Carry (*C*)
(*n*+*m*) bits Multiplication Result (*X*×*Y*)
(*n*+*m*) bits Accumulation Result (*Z*)

Figure 4.1 circuit design flow

2.2 Spanning carry look ahead adder

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is like the sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.



2.3 Modified Booth Encoder

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

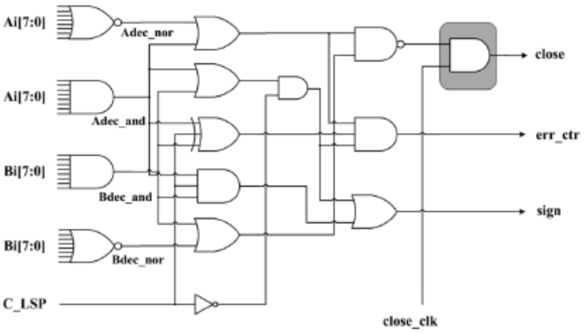


Figure 4.3 Modified booth encoder

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, only takes every second column, and multiply by ±1, ±2, or 0, to obtain the same results.

The advantage of this method is the having of the number of partial products. To Booth recode the multiplier term and consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Shows the grouping of bits from the multiplier term for use in modified booth encoding.

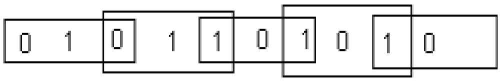
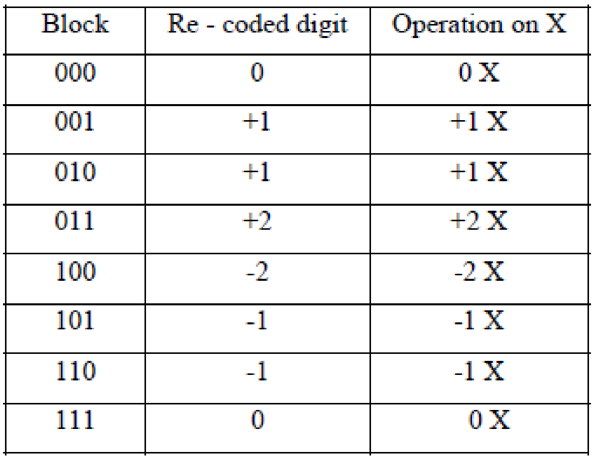


Figure 4.4 Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 4.1



Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

Table 4.1 modified booth encoder

For the partial product generation and adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits.

Computing example of Booth multiplying two numbers ”2AC9” and “006A”. The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals. According to the analysis of the multiplication .It propose the SPST-equipped modified-Booth encoder, which is controlled by a detection unit. The detection unit has one of the two operands as its input to decide whether the Booth encoder calculates redundant computations. As shown in figure 9. The latches can, respectively, freeze the inputs of MUX-4 to MUX-7 or only those of MUX-6 to MUX-7 when the PP4 to PP7 or the PP6 to PP7 are zero; to reduce the transition power dissipation. Figure 10, shows the booth partial product generation circuit. It includes AND/OR/EX-OR logic.

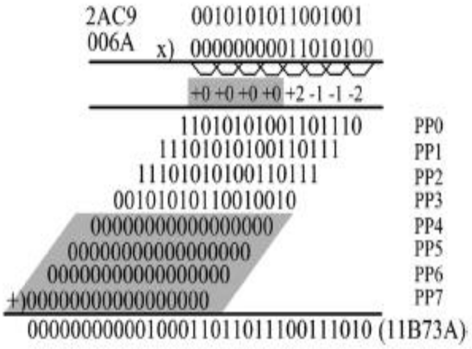


Figure.4.6 Illustration of multiplication using modified Booth encoding

The PP generator generates five candidates of the partial products, i.e., {-

2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

2.4 PARTIAL PRODUCT GENERATOR

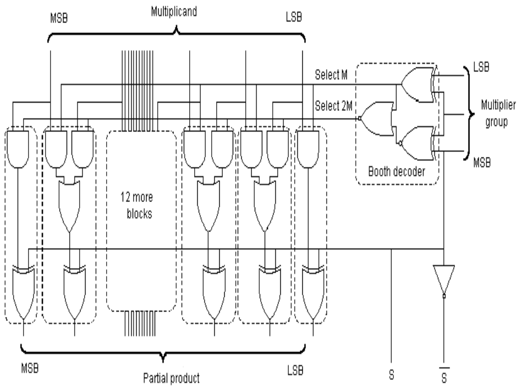


Figure 4.7 Booth partial product selector logic

The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2's complement it is negative.

The partial product is generated by doing AND between ‘a’ and ‘b’ which are a 4 bit vectors and take four bit multiplier and 4-bit multiplicand get sixteen partial products in which the first partial product is stored in ‘q’. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

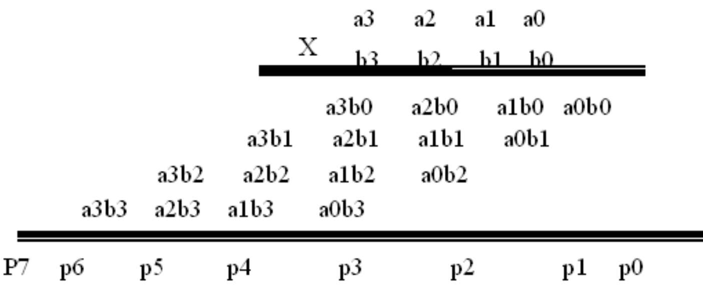


Figure 4.8 Booth partial products Generation

The multiplication second step reduces the partial products from the preceding step into two numbers while preserving the weighted sum. The sough after product P is the sum of those two numbers. The two numbers will be added during the third step The "Wallace trees" synthesis follows the Dadda's algorithm, which assures of the minimum counter number. If on top of that impose to reduce as late as (or as soon as) possible then the solution is unique. The two binary number to be added during the third step may also be seen a one number in CSA notation (2 bits per digit).

3. MODIFIED WALLACE IMPLEMENTATION

A modified Wall ace multiplier is an efficient with the hardware implementation of digital circuit As Like this multiplying two integers. Generally the Wallace multipliers uses many full adders and half addersare used for the reduction phase. To reduce the number of partial product bits in the wallece multiplier we have to use the half adders. The complexity of the multiplier is based on the numbers of half adders used in the multiplier so if we less number of half adders then complexity is reduced, a modification to the Wallace reduction is as done in which the delay is the same as for the efficient conventional Wallace reduction. The modified collection as half adders are reduced by by increase in the number of full adder In the wallece multiplication.

Reduced complexity Wallace multiplier reduction consists of three stages. First stage the N x N product matrix are rearranged in the shape of pyramid before passing to the second phase is formed and before the passing on to the second phase . During the second phase the rearranged product matrix is grouped into non-overlapping group of three as like to shown in the figure 2, in the full adder the sign bit and two sign bit will give it into next stage

$$r_{j+1}= 2[r_i/3]+r_j\text{mod}3$$

If r_j mod3 = 0, then $r_{j+1} = 2r/3$

If the value calculated from the above equation for the great as the number of rows in each stage in the second phase and the one of the second stage does not match , only then the half adder will be used. The final product of the second stage will be in the height of two bits and passed onto the third stage. The carry propagation is acgieved in the third stage.During the third stage the output of the second stage is given to the carry propagation of the achieved and generate at the final output.

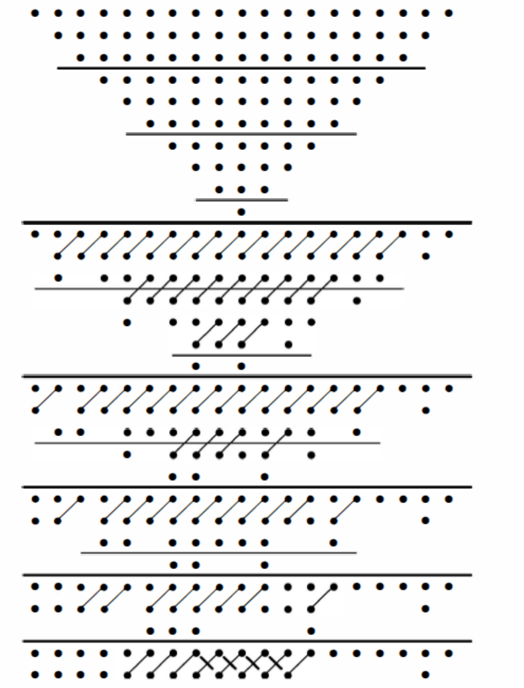


Figure.3.1: Modified Wallace 10-bit by 10-bit reduction

Thus 64 bit modified Wallace multiplier is one of the constructed and the total number of stages in the greatest second phase is 10. As per the equation the number which one of row in each of the 10 stages was calculated and the one of the use of half adders was restricted only to the 10th stage. The total number of half adders used in the achieved second phase is 6 and the total number of full adders that was used during the second phase is slightly with little bit increased that in the conventional Wallace multiplier.

Since the 64 bit modified Wallace multiplier is one of the difficult to represent, a typical l0-bit by 10-bitreduction shown in figure 2 for understanding. The one of the achieved great modified Wallace tree shows better performance in the design cycle when carry save adder is used in final stage instead of ripple carry adder. The carry save adder which is used is considered to be the critical part in themultiplier because it is responsible for the largest amount of computation.

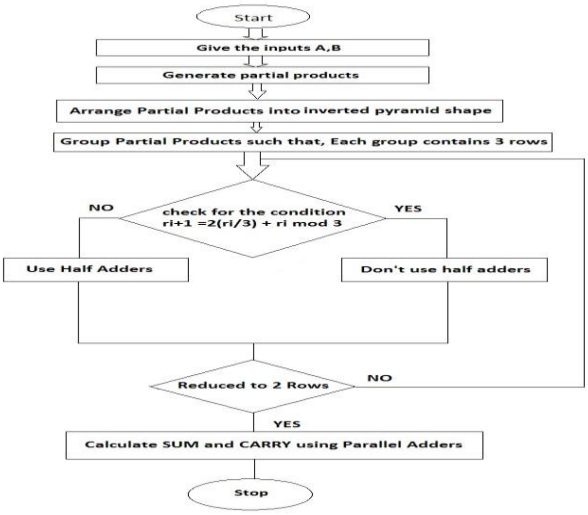

Start
Give the inputs A,B
Generate partial products
Arrange Partial Products into inverted pyramid shape
Group Partial Products such that, Each group contains 3 rows
NO
check for the condition
ri+1 =2(ri/3) + ri mod 3
YES
Use Half Adders
Don't use half adders
Reduced to 2 Rows
NO
YES
Calculate SUM and CARRY using Parallel Adders
Stop

Figure 3.2: Flow chart for Modified Wallace Multiplier:

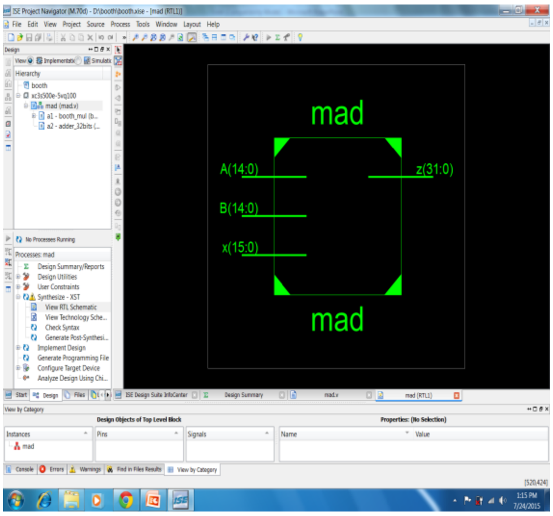


Figure : RTL schematic of modified booth recoder

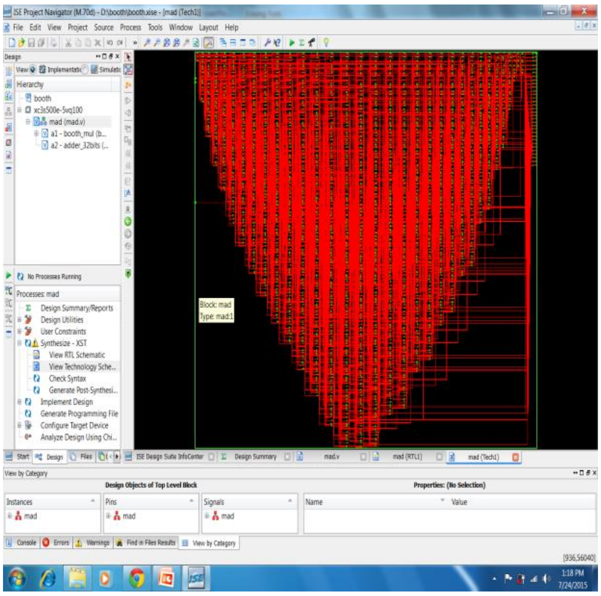
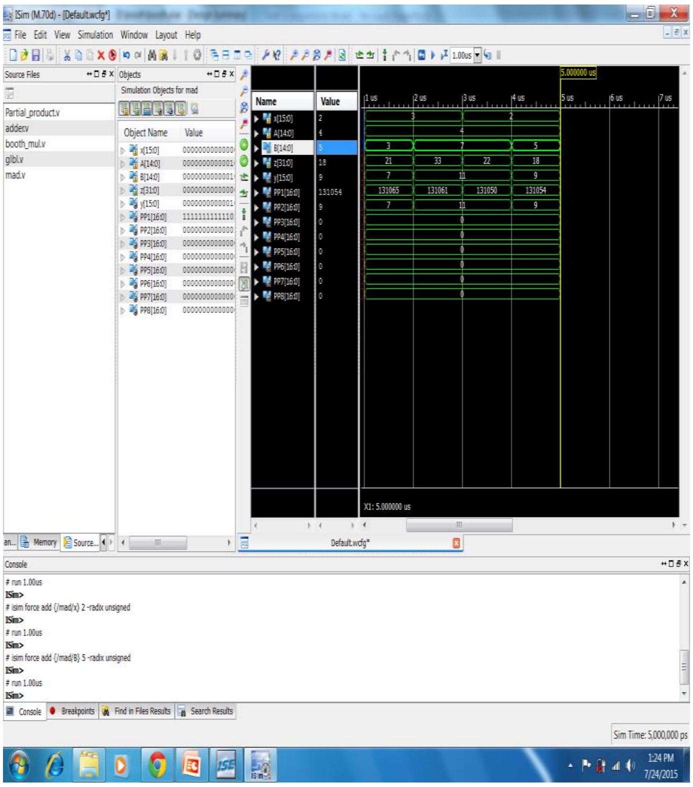


Figure: Technology schematic for modified booth multiplier

4.

RESULTS



Name	Value
x[15:0]	2
A[14:0]	4
B[14:0]	5
z[31:0]	18
y[15:0]	9
PP1[16:0]	131054
PP2[16:0]	9
PP3[16:0]	0
PP4[16:0]	0
PP5[16:0]	0
PP6[16:0]	0
PP7[16:0]	0
PP8[16:0]	0

```
# run 1.00us
ISim>
# isim force add {/mad/x} 2 -radix unsigned
ISim>
# run 1.00us
ISim>
# isim force add {/mad/B} 5 -radix unsigned
ISim>
# run 1.00us
ISim>
```

Figure: Behavioral simulation results modified booth multiplier

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

This paper focuses on optimizing the design of the Fused-Add Multiply (FAM) operator. This work presents a functional unit which is designed with multiplier-accumulator (MAC), addition, subtraction and sum of absolute difference. Compared to other circuits, the Booth multiplier has the highest operational speed and less hardware count. The basic building blocks for the unit are identified and each of the blocks is analyzed for its performance.MAC unit is designed with enable to block. Using this block, the MAC unit is constructed and calculated for the MAC unit parameters.

We propose a structured technique for the direct recoding of the sum of two numbers to its MB form. We explore three alternative designs of the proposed *S-MB* recoder and compare them to the existing ones and. The proposed recoding schemes, when they are incorporated in FAMdesigns, yield considerable performance improvements in comparison with the most efficient recoding schemes found in literature.

The presented technique explores its applications in multimedia/DSP computations, where the theoretical analysis and the realization issues are fully discussed. In this project Xilinx-ISE tool is used for logical verification, synthesizing performing placing & routing operation for system verification.

6.2 FUTURE SCOPE

In future it can be extended to floating point numbers also with the supportive EDA tools. By using transistor level implementation for the carry save logic the design reduces the total area required compared to gate level designs. There is chance to improve the speed somewhat more by changing architecture.

REFERENCES

[1] Soojin Kim and Kyeongsoon Cho “Design of High-speed Modified Booth Multipliers Operating at GHz Ranges” World Academy of Science, Engineering and Technology 61 2010.

[2] Magnus Sjalander and Per Larson-Edefors. “The Case for HPM-Based Baugh-Wooley Multipliers,” Chalmers University of Technology,Sweden, March 2008.

[3] Z Haung and M D Ercegovac, “High performance Low Power left to right array multiplier design” IEEE rans.Computer, vol 64 no3, page 272-283 Mar 2006.

[4] Aswathy Sudhakar, and D. Gokila, “Run-Time configurable Pipelined Modified Baugh-Wooley Multipliers,” Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 223–236.

[5] Myoung-Cheol Shin, Se-Hyeon Kang, and In-Cheol Park, "An Area-Efficient Iterative Modified-Booth Multiplier Based on Self-Timed Clocking," Industry, and Energy through the project System IC 2010, and by IC Design Education Center (IDEC).